Wagonblast, Gregory
CS2510

Final Project: Wake-Word Detection for Embedded Device (Arduino Nano 33 BLE Sense Lite) with LLM Integration

1. Introduction

As voice-activated technologies become increasingly integrated into modern computing systems, the need for low-power and responsive voice interfaces on embedded devices has become increasingly important. Having a reliable voice-interface as well as the efficiency of a low-power device with an energy cost of below 1 mW is desirable. The majority of modern day use cases are solved through the smartphone interface, but there remains many potential use cases that could be solved with affordable, efficient, and reliable embedded devices. This project explores a distributed system architecture where an embedded device performs wake-word detection and communicates through the Bluetooth **[1]** protocol with a more powerful server that handles complex processing.

This complex processing consists of the server handling the initialization of a color-word association game with the embedded client, prompting a large language model (LLM), as well as the game logic. This color-word association game is as follows:

- The game consists of n-rounds (five in my implementation).
- Each round:
  - ✓ Gemini **[2]**, the LLM, chooses a random color from {"green", "red", "blue"}.
  - ✓ This color is displayed on an LED on the embedded client.
  - ✓ The user speaks a corresponding word from {"yes", "no", "unknown"} into the embedded client.
  - ✓ The user's spoken response is transmitted to the server.
  - ✓ The server processes the response as correct or incorrect.
- After n-rounds, the game ends.
- The user's final score is displayed in a console window.

The initial plan for this project called for using the Arducam Pico4ML TinyML Dev Kit, but the final implementation was completed on the Arduino Nano 33 BLE Sense Lite due to improved compatibility with available tools and Bluetooth communication libraries. Although this game may not be practical, it showcases the combination of embedded intelligence and distributed communication, a fundamental concept in distributed operating systems, while working within the constraints of low-power microcontrollers. This final project retains the fundamental concepts that were meant to be addressed in the initial plan while allowing for some flexibility in the specific implementation.

Wagonblast, Gregory
CS2510

2. Technique

2a. System Overview and Architecture

The final system consists of the following components:

- **Embedded Client (Arduino Nano 33 BLE Sense Lite)**
  The Arduino performs on-device wake-word detection using a TensorFlow Lite model. Upon detecting the wake-word, it initiates a Bluetooth connection to the server.
- **Server (Laptop/PC)**
  Through the use of a python script, the server listens for BLE connections from the Arduino. When a connection is established, the server initiates a color-word association game and handles game logic, including prompting the LLM to provide random colors.
- **LLM (Gemini)**
  The Gemini model is used on the server to dynamically generate color prompts for the game. This creates a simple but engaging interactive experience that demonstrates the server's ability to handle complex processing tasks.

This setup reflects a client-server model where the embedded device acts as a thin client, handling lightweight tasks locally while offloading more complex operations to the server.

2b. Wake-Word Detection on the Embedded Device

Wake-word detection was implemented using TensorFlow Lite for microcontrollers on the Arduino Nano 33 BLE Sense Lite, which includes a built-in microphone and sufficient processing power for small neural network models.

The steps for wake-word detection included **[3] [4]**:

- *Main loop*: The application runs in a continuous loop, executing commands as fast as the microcontroller can run them.
- *Audio provider*: Captures raw audio data from the microphone.
- *Feature provider*: Converts raw audio data into the spectrogram format that the model requires.
- *TF Lite interpreter*: Transforms the input spectrogram into a set of probabilities.
- *Model*: This is a data array that is run by the interpreter.
- *Command recognizer*: Aggregates results and determines whether, on average, a known word was heard.
- *Command responder*: Uses the device's output capabilities to handle user interaction.

The model was optimized to fit within the Arduino's limited memory constraints, 256KB of SRAM for runtime operations and 1MB of Flash for storing the model and program code. This achieved acceptable performance, with minimal latency, and a relatively low false-positive rate during testing.

2c. Bluetooth Communication and Client-Server Interaction

Following successful wake-word detection, the Arduino connects to the server over Bluetooth Low Energy (BLE). The server was developed using Python and the bleak library **[5]**, which enables reliable and asynchronous communication with the embedded device.

The client-server communication protocol included the following steps:

1. **Activation**: The embedded client sends a trigger message upon wake-word detection.
2. **Game Initialization**: The server requests a set of random colors from the Gemini LLM.
3. **Game Execution**: The server sends one color each round to the client, prompting the user to say the corresponding word.
4. **Verification**: The corresponding word is sent to the server for verification.
5. **Feedback**: The server responds with feedback ("Correct" or "Incorrect") to the console window and continues the game loop.

This interaction loop demonstrates distributed task allocation, offloading computationally expensive and dynamic tasks to the server while the embedded device remains reactive and efficient.

2d. LLM Integration and Game Logic

The LLM was integrated into the system using the Gemini API. Instead of processing arbitrary user input, the LLM was tasked with generating randomized color prompts to power a voice-controlled color association game. This constrained use of the LLM was sufficient to demonstrate:

- Dynamic prompt generation.
- Natural language usage in embedded applications.
- The effectiveness of combining embedded systems with high-level AI systems in a distributed architecture.

This gameplay loop, initiated by a wake-word and executed through a combination of BLE communication and LLM generation, offered an interesting demonstration of natural language interfaces for embedded systems.

3. Results and Evaluation

The final system achieved several key milestones:

- **Wake-word detection** was successfully implemented on-device with low latency, approximately less than 1 ms.
- **Distributed communication** was reliable, with round-trip communication times approximately less than 10ms.
- **LLM interaction** provided randomized game content with response times approximately less than 1ms per color.
- **Color-word association game** was fully functional, though limited to a closed domain due to hardware constraints.
- **Power use** was low-power **[6]**. Estimated 2.15 mW for microphone, 2.61 mW for LEDs, 33 mW for active Bluetooth, 1 mW when wake-word inactive, and 18.15 mW when inference is active. Totaling an approximate 56.91 mW power usage.

Performance Highlights:

Metric	**Result**
Wake-word detection latency	~ less than 1 ms
BLE communication latency	~ less than 10 ms round-trip
LLM response latency (Gemini)	~ less than 1 ms
False wake trigger rate	common
Power Usage	~ 56.91 mW

Limitations:

- Full voice-to-text command processing was not implemented.
- Bluetooth pairing could be inconsistent due to OS limitations.
- Wake-word detection subject to common false wake triggers.
- The proposed teacher-student model (LLM compression for client use) was not completed due to time constraints **[7]**.

Wagonblast, Gregory
CS2510

4. Conclusion and Future Work

This project successfully demonstrated a functional distributed system using embedded machine learning and Bluetooth-based client-server communication. The integration of a wake-word detection model on a microcontroller with server-side language processing created a responsive, interactive system capable of playing a simple voice-controlled game.

The use of the Arduino Nano 33 BLE Sense Lite proved effective for real-time interaction, and the system architecture provides a foundation for future expansion. Key directions for future work include:

- Extending the wake-word system to support more commands.
- Implementing full voice-to-text command processing.
- Introducing the teacher-student model to transfer knowledge from a server-hosted LLM to an on-device student model.

This project highlights the power and potential of combining TinyML/EmbeddedAI and distributed operating system principles to create intelligent, interactive embedded applications.

References

[1] **Google, "Generating content," *Gemini API Documentation*, Apr. 11, 2025. [Online]. Available: https://ai.google.dev/api/generate-content. [Accessed: Apr. 19, 2025].**

[2] **Bluetooth SIG, *Assigned Numbers*, Version Date: April 9, 2025. [Online]. Available: https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Assigned_Numbers/out/en/Assigned_Numbers.pdf. [Accessed: Apr. 19, 2025].**

[3] **P. Warden and D. Situnayake, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. Sebastopol, CA: O'Reilly Media, 2019.**

[4] **TensorFlow Authors, "*TensorFlow Lite for Microcontrollers*," GitHub repository, https://github.com/tensorflow/tensorflow/tree/be4f6874533d78f662d9777b66abe3cdde98f901/tensorflow/lite/experimental/micro, accessed Apr. 19, 2025.**

[5] **H. Blidh, "Bleak: Bluetooth Low Energy platform Agnostic Klient," Read the Docs. [Online]. Available: https://bleak.readthedocs.io/en/latest/. [Accessed: Apr. 19, 2025].**

[6] **Arduino, "Nano 33 BLE Sense," Arduino Documentation. [Online]. Available: https://docs.arduino.cc/hardware/nano-33-ble-sense/#compatibility. [Accessed: Apr. 19, 2025].**

[7] **C. Hu *et al.*, "Teacher-Student Architecture for Knowledge Distillation: A Survey," *arXiv preprint* arXiv:2308.04268, Aug. 2023. [Online]. Available: https://arxiv.org/abs/2308.04268**